2.15 COMMAND CHAIN HIERARCHY

The U.S. military defines command and control as the exercise of authority and direction by a properly designated commander over assigned and attached forces in the accomplishment of the mission. Command and control functions are performed through an arrangement of personnel, equipment, communications, facilities, and procedures employed by a commander in planning, directing, coordinating, and controlling forces and operations in accomplishment of the mission.

This functional element describes the command chain hierarchy of a side of a conflict. In Suppressor this is known as a command chain. The selection of players along with their arrangement in the SDB defines the command chain hierarchy.

2.15.1 Functional Element Design Requirements

a. The model will require the user to create at least one command chain for each side in a scenario, and to assign each player to at least one command chain. A player will also be allowed to be on multiple command chains, possibly on different sides. The assignment of a player to a command chain will imply only that the player has friendly perceptions of its immediate commanders (if any) and immediate subordinates (if any) on the chain.

b. The model will provide a capability for dynamic reorganization of a command chain by allowing alternate commanders. The alternative commander will initially be a subordinate of the original commander and will assume lethal assignment responsibilities when the original commander is destroyed or is unable to communicate with subordinates for some user-defined period of time.

2.15.2 Functional Element Design Approach

In Suppressor, command chains define the C^3 hierarchy of players. A player must belong to at least one command chain, but can belong to several.

When multiple chains are used to describe a side, frequently it is the case that the user wishes to define a different hierarchy for different functions. One chain might define the command hierarchy to be followed for making target assignments to subordinates, while another might define the structure for relaying intelligence information about targets.

Design Element 15-1: Command Chain

To specify the participating player in a command chain as well as his relative position in the hierarchy, one uses the PLAYER: input item in the SDB. The PLAYER: item defines the specific player id and name and also his level within the chain. A player given a level of 1 is at the top of the hierarchy. Successive levels are numbered 2, 3, etc.

The order that the PLAYER: data items appear in the command chain is significant. The PLAYER: items are placed in the chain in a preorder (also known as prefix) manner. That is, the vertical (parent-child) relationships are specified before the horizontal (sibling) relationships. A few short examples follow:

```
PLAYER: 1 hq            LEVEL: 1
PLAYER: 2 sam_regiment  LEVEL: 2
PLAYER: 3 sam_battery   LEVEL: 3
PLAYER: 4 sam_regiment  LEVEL: 2
PLAYER: 5 sam_battery   LEVEL: 3
```

In this example, the "hq" player is at the top. He has two immediate subordinates (players 2 and 4), who each have one subordinate battery (players 3 and 5, respectively).

```
PLAYER: 1 hq            LEVEL: 1
PLAYER: 2 sam_regiment  LEVEL: 2
PLAYER: 4 sam_regiment  LEVEL: 2
PLAYER: 3 sam_battery   LEVEL: 3
PLAYER: 5 sam_battery   LEVEL: 3
```

In this example, the "hq" is again the top commander with two immediate subordinates (players 2 and 4). However, both battery players are now subordinate to player 4.

Design Element 15-2: Dynamic Reorganization

This functional element handles the situations where the pre-scenario command and control hierarchy dynamically changes during scenario execution.

In Suppressor, there are several events which may alter somewhat the command and control hierarchy established in the SDB. These are:

a. A commander is killed and an alternate commander is specified (via the SDB input ALT-CMDRS).

b. A subordinate is unable to establish contact with his commanders for a certain period of time.

c. A commander decides to tell a subordinate to go autonomous or give up autonomy.

In the TDB, there is an optional input ASG-CMD-CHAIN which lists all command chains through which the player can accept an assignment. When the player has lost communication with the commanders on all of the named chains for a period exceeding the input COMM-LOSS-DECENT-TIME, he will decentralize control to himself. This causes the player to go autonomous, which implies that he no longer has to have permission from a commander to engage a target.

In the TDB RESOURCE-ALLOCATION procedures, a commander can evaluate criteria which may cause his subordinate(s) to go autonomous (a GUNS-FREE order) or give up autonomy (a GUNS-TIGHT order). Such a message will reset the internal value (ENGAGE MODE-OF-CONTROL) in the subordinate's data base, which he checks during his local decisions whether to engage and/or fire at a target.

2.15.3 Functional Element Software Design

Command Chain Module Design

Command chains are created in routine VARSUB based on the SDB level: data item. Checks are made to see if the LEVEL: specified is more than one level different than the previously processed player. Then, the player and command chain data structures are allocated.

```
*when PLAYER:  input:
      *signal an error if two or more levels down;
      *loop, while pointing to level lower or at input level:
         *move ptr one level up, increment difference;
         *move ptr one level up, increment difference;
      *end of loop for looking for previous level.
      *allocate storage for command chain block;
      *loop, while block not found, blocks left on list:
         *set ptr when block found;
      *end of loop for block not found, blocks left.
      *when player block does not exist:
         *allocate player and store side code;
         *store unit type, organizational number;
         *store disaggregation flag, if necessary;
         *increment unique player id and store in player;
         *add SDB player to list;
         *allocate storage for SDB orders and store pointer;
         *add interaction key to list if not disaggregation;
         *store SDB values in the interaction key;
         *allocate a central directory;
      *end of test for player block.
      *add command chain node to player's list;
   *when input not valid player name, notify user;
```

Dynamic Reorganization Module Design

For case a, when a commander is killed and an alternate commander has been specified, the routine ALTCMD performs the following:

```
*loop over alternate commanders in priority order:
   *search for alt cmdr that's alive:
*end loop over alt cmdr list.
*when alternate commander was found:
   *when player is new commander:
      *report the seizing of control;
      *loop over dead cmdr's subs:
         *when sub is alive and not new cmdr:
            *add to perceived sub list;
            *no perceived sub list is included for new sub;
            *loop over perceived nets;
               *check if sub already on perceived net;
               *check if sub on actual net;
               *if sub is not on the perceived net:
                  *if sub found on actual net:
                     *load system status;
                     *add communication net buffer;
                  *end of test for sub on actual net.
               *end of test for sub on perceived net.
               *check if new cmdr on sub perceived net;
                  *if cmdr is not on subs perceived net;
                     *add alt cmdr to subs perceived net;
                     *add communications net buffer;
                  *end of test for cmdr not on net.
```

```
                  *end of test for cmdr on sub perceived net.
               *end of loop over perceived nets.
               *create new perceived commander for each sub;
            *end test for new cmdr on sub list.
         *end loop over new subs.
      *otherwise, someone else is new commander:
         *change perception of cmdr;
      *end test for alt cmdr identity.
   *end test for alt cmdr found.
   *when no alt cmdr found or this player is new commander:
      *loop, for each assignment command chain:
         *set flag if chain of dead commander is an asg cmd chain;
      *end of loop for each assignment command chain:
      *when dead commander is on an assignment command chain:
         *when current mode of control not himself:
            *loop, for each assignment command chain:
               *when this player has a commander in the chain:
                  *set alive flag if commander is alive;
               *end of test for having a commander in the chain.
            *end of loop for each assignment command chain.
         *end of test for current mode of control not himself.
         *when no living commanders on any assign command chain:
            *invoke logic to properly assume control;
            *when not thinking of emcon and tactics exist:
               *add to emcon to pending queue, set evaluation flag;
            *end of test for EMCON tactics.
         *end of test for no living commanders on any asg cmd chain.
      *end of test for dead commander on an assignment command chain.
      *remove perception of cmdr;
   *end test for no alt cmdr or player is new cmdr.
```

For case b, when the subordinate decentralizes due to inability to talk with his commanders, routine SEEFUZ performs the following:

```
 *when not own mode of control and comm. loss time specified:
    *loop, for each specified assign command chain:
       *when recipient is commander on this chain:
          *reset comm. loss time if first failed attempt;
       *end of test for recip. is commander on this chain.
       *set flag to not decentralize if no comm. loss or
       * comm. loss does not exceed limit;
    *end of loop, for each specified assign command chain.
    *when recip. is a commander on an asg. cmd chain and comm
    * loss decent time exceeded for all asg. cmd. chains:
       *decentralize to your mode of control;
       *write "lost contact with" incident;
       *when not thinking of emcon and tactics exist:
          *add to emcon to pending queue, set eval flag;
       *end of test for EMCON tactics.
    *end of test for comm. loss decent time exceeded for all
    * asg. cmd. chains.
 *end of test for mode of control and comm. loss time.
```

For case c, when the commander makes a decision to allow the subordinate to go autonomous or to take back the autonomy, the routine EFESUB performs the following:

```
*when mode-of-control change:
   *when subordinate was selected:
      *allocate MOC message;
      *invoke logic to send MOC change message;
      *search for resource type on choose-from list;
      *invoke logic to implement possible with-cuing phrases;
   *end of test for subordinate selected.
```

2.15.4 Assumptions and Limitations

A player designated as the alternate commander must be a subordinate of the original commander.

2.15.5 Known Problems or Anomalies

None.